GROKKING THE JAVA INTERVIEW

JAVIN PAUL @JAVINPAUL

Table Of Contents

PREFACE
Object Oriented Programming Interview Questions
What is method overloading in OOP or Java?
What is the method overriding in OOP or Java?
What is the method of hiding in Java?
Is Java a pure object-oriented language? if not why?
What are the rules of method overloading and overriding in Java?
The difference between method overloading and overriding?
Can we overload a static method in Java?
Can we override the static method in Java?
Can we prevent overriding a method without using the final modifier?
Can we override a private method in Java?
What is the covariant method overriding in Java?
Can we change the return type of method to subclass while overriding?

Can we change the argument list of an overriding method?
Can we override a method that throws runtime exception without throws clause?
How do you call a superclass version of an overriding method in a subclass?
Can we override a non-static method as static in Java?
Can we override the final method in Java?
Can we have a non-abstract method inside an interface?
What is the default method of Java 8?
What is an abstract class in Java?
What is an interface in Java? What is the real user of an interface?
The difference between Abstract class and interface?
Can we make a class abstract without an abstract method?
Can we make a class both final and abstract at the same time?
Can we overload or override the main method in Java?

What is the difference between Polymorphism, Overloading, and Overriding?
Can an interface extend more than one interface in Java?
Can a class extend more than one class in Java?
What is the difference between abstraction and polymorphism in Java?
What problem is solved by the Strategy pattern in Java?
Which OOP concept Decorator design Pattern is based upon?
When to use the Singleton design pattern in Java?
What is the difference between State and Strategy Patterns?
What is the difference between Association, Aggregation, and Composition in OOP?
What is the difference between Decorator, Proxy and Adapter pattern in Java?
What is the 5 objects oriented design principle from SOLID?
What is the difference between Composition and Inheritance in OOP?

PREFACE

Cracking a Java Interview is not easy and one of the main reason for that is Java is very vast. There are lot of concepts and APIs to master to become a decent Java developer. Many people who are good at general topics like Data Structure and Algorithms, System Design, SQL and Database fail to crack the Java interview because the don't spend time to learn the Core Java concepts and essential APIs and packages like Java Collection Framework, Multithreading, JVM Internals, JDBC, Design Patterns and Object Oriented Programming.

This book aims to fill that gap and introduce you will classical Java interview questions from these topics. By going through these questions and topic you will not only expand your knowledge but also get ready for your Next Java interview. If you are preparing for Java interviews then I highly recommend you go through these questions before your telephonic or face-to-face interviews, you will not only gain confidence and knowledge to answer the question but also learn how to drive Java interview in your favor. This is the single most important tip I can give you as a Java developer. Always, remember, your answers drive interviews, and these questions will show you how to drive Interviewer to your strong areas. All the best for Java interview and if you have any questions or feedback you can always contact me on twitter javinpaul or comment on my blogs Javarevisited and Java67.

OBJECT ORIENTED PROGRAMMING INTERVIEW QUESTIONS

Java is an object-oriented programming language and you will see a lot of object-oriented programming concept questions on Java interviews. The classic questions like the difference between an interface and abstract class are always there but from the last couple of years more sophisticated questions based upon advanced design principles and patterns are also asked to check OOP knowledge of the candidate. Though, Object-oriented programming questions are more popular on Java interviews for 1 to 3 years experienced programmers. It makes sense as well, as these are the programmers who must know the OOP basics like Abstraction, Inheritance, Composition, Class, Object, Interface, Encapsulation, etc.

If you look for Java interview questions for 2 to 4 years experienced programmer, you will find lots of questions based upon OOP fundamentals like Inheritance and Encapsulation but as you gain more experience, you will see questions based on object-oriented analysis and design e.g. code a vending design machine or implement a coffeemaker in Java.

These questions are more difficult and require not only a true understanding of OOP fundamentals but also about SOLID design principles and patterns.

In this section, I am going to share with you some OOPS concept based Java interview questions that I have collected from friends and colleagues and they have seen in various Java interviews on different companies. They are mostly asked at first a few rounds like on screening round or on the telephonic round.

If you are a senior Java developer then you already know answers to this question and I suggest you practice more on object-oriented analysis and design skill i.e. how to do code against a specification. If you are fresher and junior Java developer with 2 to 3 years experience then you must revise these questions, learn if you don't know to do well on your Java Job interviews.

What is method overloading in OOP or Java?

It's one of the oldest OOPS concept questions, I have seen it 10 years ago and still sees it now. When we have multiple methods with the same name but different functionality then it's called method overloading. For example. System.out.println() is overloaded as we have a 6 or 7 println() method each accepting a different type of parameter.

What is the method overriding in OOP or Java?

It's one of the magic of object-oriented programming where the method is chose based upon an object at runtime. In order for method overriding, we need Inheritance and Polymorphism, as we need a method with the same signature in both superclass and subclass. A call to such a method is resolved at runtime depending upon the actual object and not the type o variable.

What is the method of hiding in Java?

When you declare two static methods with same name and signature in both superclass and subclass then they hide each other i.e. a call to the method in the subclass will call the static method declared in that class and a call to the same method is superclass is resolved to the static method declared in the superclass.

Is Java a pure object-oriented language? if not why?

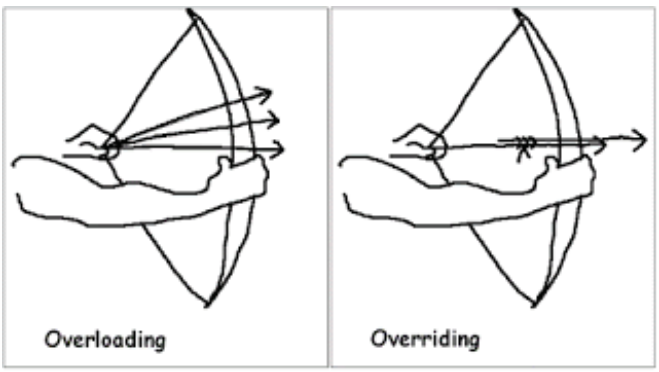
Java is not a pure object-oriented programming language e.g. there are many things you can do without objects e.g. static methods. Also, primitive variables are not objects in Java.

What are the rules of method overloading and overriding in Java?

One of the most important rules of method overloading in Java is that the method signature should be different i.e. either the number of arguments or the type of arguments. Simply changing the return type of two methods will not result in overloading, instead, the compiler will throw an error. On the other hand, method overriding has more rules e.g. name and return type must be the same, method signature should also be the same, the overloaded method cannot throw a higher exception, etc.

The difference between method overloading and overriding?

Several differences but the most important one is that method overloading is resolved at compile-time and method overriding is resolved at runtime. The compiler only used the class information for method overloading, but it needs to know the object to resolved overridden method calls. This diagram explains the difference quite well, though:



Can we overload a static method in Java?

Yes, you can overload a static method in Java. You can declare as many static methods of the same name as you wish provided all of them have different method signatures.

Can we override the static method in Java?

No, you cannot override a static method because it's not bounded to an object. Instead, static methods belong to a class and resolved at compile time using the type of reference variable. But, Yes, you can declare the same static method in a subclass, that will result in method hiding i.e. if you use the reference variable of type subclass then new method will be called, but if you use the reference variable of superclass than old method will be called.

Can we prevent overriding a method without using the final modifier?

Yes, you can prevent the method overriding in Java without using the final modifier. In fact, there are several ways to accomplish it e.g. you can mark the method private or static, those cannot be overridden.

Can we override a private method in Java?

No, you cannot. Since the private method is only accessible and visible inside the class they are declared, it's not possible to override them in subclasses. Though, you can override them inside the inner class as they are accessible there.

What is the covariant method overriding in Java?

In the covariant method overriding, the overriding method can return the subclass of the object returned by the original or overridden method. This concept was introduced in Java 1.5 (Tiger) version and it's very helpful in case the original method is returning general type like Object class, because, then by using the covariant method overriding you can return a more suitable object and prevent client-side typecasting. One of the practical use of this concept is when you override the clone() method in Java.

Can we change the return type of method to subclass while overriding?

Yes, you can, but only from Java 5 onward. This feature is known as covariant method overriding and it was introduced in JDK 5 release. This is immensely helpful if the original method return super-class like clone() method return java.lang.Object. By using this, you can directly return the actual type, preventing client-side type-casting of the result.

Can we change the argument list of an overriding method?

No, you cannot. The argument list is part of the method signature and both overriding and overridden methods must have the same signature.

Can we override a method that throws runtime exception without throws clause?

Yes, there is no restriction on unchecked exceptions while overriding. On the other hand, in the case of checked exception, an overriding exception cannot throw a checked exception which comes higher in type hierarchy e.g. if the original method is throwing IOException than the overriding method cannot throw java.lang.Exception or java.lang.Throwable.

How do you call a superclass version of an overriding method in a subclass?

You can call a superclass version of an overriding method in the subclass by using super keyword. For example to call the toString() method from java.lang.Object class, you can call super.toString().

Can we override a non-static method as static in Java?

Yes, you can override the non-static method in Java, no problem on them but it should not be private or final :)

Can we override the final method in Java?

No, you cannot override a final method in Java, the final keyword with the method is to prevent method overriding. You use the final when you don't want subclass changing the logic of your method by overriding it due to security reasons. This is why the String class is final in Java. This concept is also used

in the template design patterns where the template method is made final to prevent overriding.

Can we have a non-abstract method inside an interface?

From Java 8 onward you can have a non-abstract method inside interface, prior to that it was not allowed as all method was implicitly public abstract. From JDK 8, you can add static and default methods inside an interface.

What is the default method of Java 8?

The default method, also known as the extension method is new types of the method which you can add on the interface now. These method has implementation and intended to be used by default. By using this method, JDK 8 managed to provide common functionality related to lambda expression and stream API without breaking all the clients which implement their interfaces. If you look at Java 8 API documentation you will find several useful default methods on key Java interface like Iterator, Map, etc.

What is an abstract class in Java?

An abstract class is a class that is incomplete. You cannot create an instance of an abstract class in Java. They are provided to define default behavior and ensured that client of that class should adore to those contract which is defined inside the abstract class. In order to use it, you must extend and implement their abstract methods. BTW, in Java, a class can be abstract

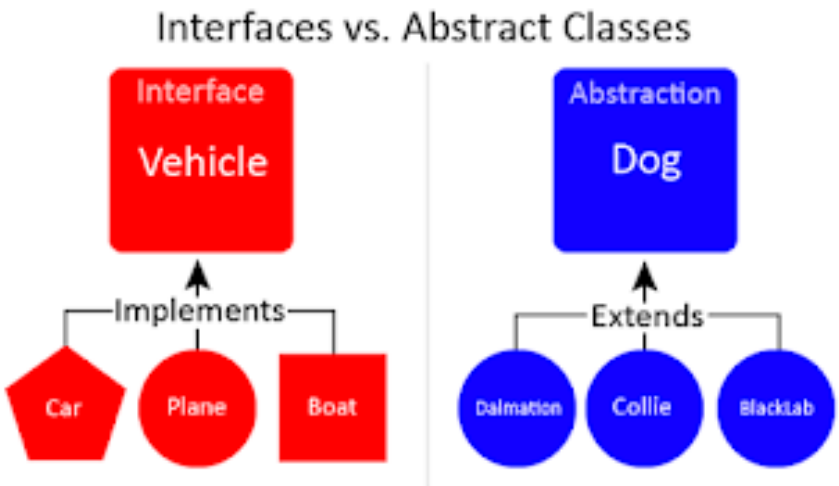
without specifying any abstract method.

What is an interface in Java? What is the real user of an interface?

Like an abstract class, the interface is also there to specify the contract of an API. It supports the OOP abstraction concept as it defines only abstract behavior. It will tell that your program will give output but how is left to implementors. The real use of the interface to define types to leverage Polymorphism.

The difference between Abstract class and interface?

In Java, the key difference is that abstract class can contain a non-abstract method but the interface cannot, but from Java 8 onward interface can also contain static and default methods that are non-abstract.



Can we make a class abstract without an abstract method?

Yes, just add abstract keyword on the class definition and your class will become abstract.

Can we make a class both final and abstract at the same time?

No, you cannot apply both final and abstract keyword at the class at the same time because they are exactly opposite of each other. A final class in Java cannot be extended and you cannot use an abstract class without extending and make it a concrete class. As per Java specification, the compiler will throw an error if you try to make a class abstract and final at the same time.

Can we overload or override the main method in Java?

No, since main() is a static method, you can only overload it, you cannot override it because the static method is resolved at compile time without needing object information hence we cannot override the main method in Java.

What is the difference between Polymorphism, Overloading, and Overriding?

This is a slight tricky OOP concept question because Polymorphism is the real concept behind on both Overloading and Overriding. Overloading is compiled time Polymorphism and Overriding are Runtime

Polymorphism.

Can an interface extend more than one interface in Java?

Yes, an interface can extend more than one interface in Java, it's perfectly valid.

Can a class extend more than one class in Java?

No, a class can only extend another class because Java doesn't support multiple inheritances but yes, it can implement multiple interfaces.

What is the difference between abstraction and polymorphism in Java?

Abstraction generalizes the concept and Polymorphism allows you to use different implementation without changing your code. This diagram explains the abstraction quite well, though:

Object-Oriented design principle and pattern Interview Questions

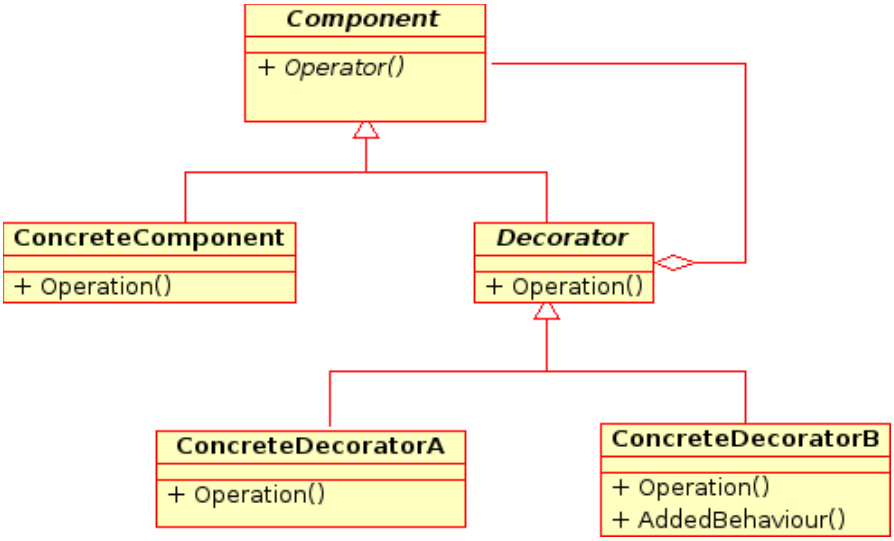
Now let's see some OOPS concept questions based on the SOLID design principles and GOF design patterns that take advantage of the OOPS concept discussed here.

What problem is solved by the Strategy pattern in Java?

Strategy pattern allows you to introduce a new algorithm or new strategy without changing the code which uses that algorithm. For example, the Collections.sort() method which sorts the list of the object uses the Strategy pattern to compare objects. Since every object uses a different comparison strategy you can compare various objects differently without changing the sort method.

Which OOP concept Decorator design Pattern is based upon?

The decorator pattern takes advantage of Composition to provide new features without modifying the original class. A very good to-the-point question for the telephonic round. This is quite clear from the UML diagram of the Decorator pattern, as you can see the Component is associated with a Decorator.



When to use the Singleton design pattern in Java?

When you need just one instance of a class and want that to be globally available then you can use the Singleton pattern. It's not free of cost though because it increases the coupling between classes and makes them hard to test. This is one of the oldest design pattern questions from Java interviews.

What is the difference between State and Strategy Patterns?

Though the structure or class diagram of State and Strategy pattern is the same, their intent is completely different. The state pattern is used to do something specific depending upon state while Strategy allows you to switch between algorithms without changing the code which uses it.

What is the difference between Association, Aggregation, and Composition in OOP?

When an object is related to another object is called association. It has two forms, aggregation, and composition. the former is the loose form of association where the related object can survive individually while later is a stronger form of association where a related object cannot survive individually. For example, the city is an aggregation of people but is the composition of body parts.

What is the difference between Decorator, Proxy and Adapter pattern in Java?

Again they look similar because their structure or class diagram is very similar but their intent is quite different. The Decorator adds additional functionality without touching the class, Proxy provides access control and Adapter is used to make two incompatible interfaces work together.

What is the 5 objects oriented design principle from SOLID?

SOLID is the term given by Uncle Bob in his classic book, the Clean Code, one of the must-read books for programmers. In SOLID each character stands for one design principle:

- S for Single Responsibility Principle
- O for Open closed design principle
- L for Liskov substitution principle
- I for Interface segregation principle
- D for Dependency inversion principle

What is the difference between Composition and Inheritance in OOP?

This is another great OOPS concept question because it tests what matters, both of them are very important from a class design perspective. Though both Composition and Inheritance allows you to reuse code, formerly is more flexible than later. Composition allows the class to get an additional feature at runtime, but Inheritance is static. You can not change the feature at runtime by substitution of a new implementation.

That's all about in this list of object-oriented programming or OOPS concept interview questions. We have seen questions from various OOPS concepts like Abstraction, Encapsulation, Inheritance, Composition, Aggregation, Association, Class, Object,

and Interface, etc.

We have also seen questions from Object-oriented design principles also known as SOLID principles and GOF design patterns like Strategy, State, and Factory, which are based upon both the object-oriented programming concept and OOP design principle.

But, as I said, if you are a senior Java developer then you focus more on object-oriented analysis and design and learn how to code against a requirement using all your OOP knowledge. You can also read Cracking the Coding Interview, 6th Edition to for more Object Oriented Programming questions.